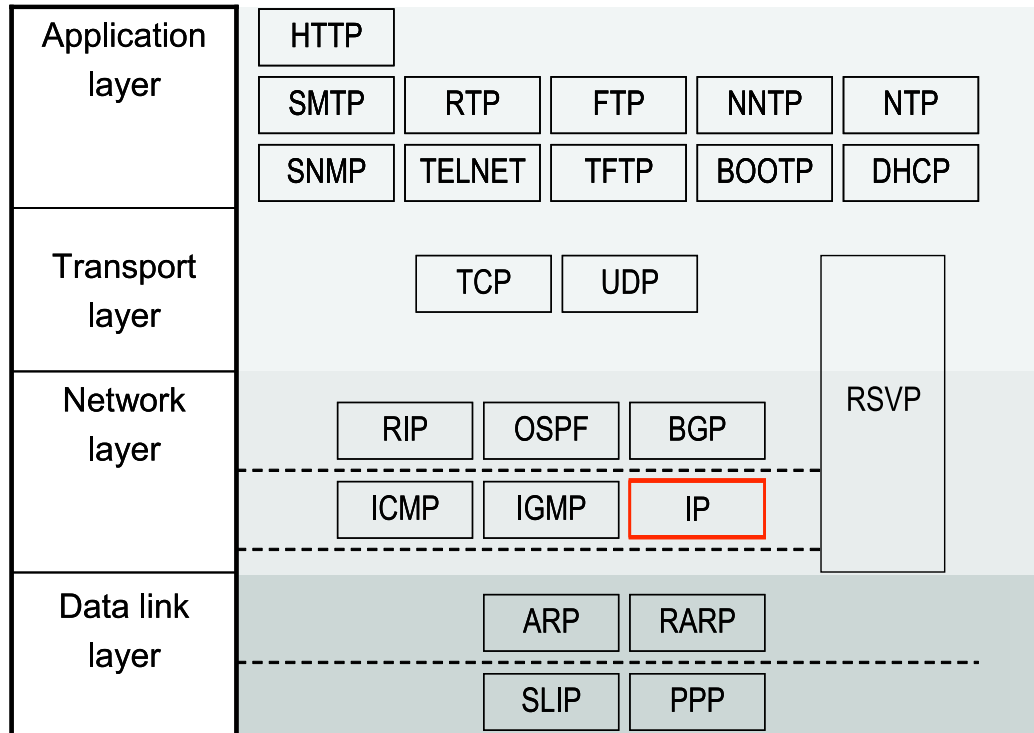


Rechnernetze

Allgemeine Konzepte:

Internet Modell:



Netze:

Adressklassen:

- Klasse: Vorgabe der Länge der Netz-ID und eines Netzpräfix
- an Präfix kann die Netzklasse abgelesen werden

Class A 0 Netz (7 Bits) Host (24 Bits)

Class B 10 Netz (14 Bits) Host (16 Bits)

Class C 110 Netz (21 Bits) Host (8 Bits)

- Beispiele

Class A 00001010 00000000 00000000 00100000 = 10.0.0.32

Class B 10000010 00000011 00000011 00111100 = 130.3.3.60

Class		Netz ID	Host ID	Subnetze	Hosts
A	$p < 127$	p	q.r.s	126	16777214
B	$128 \leq p \leq 191$	p.q	r.s	16382	65534
C	$192 \leq p \leq 223$	p.q.r	s	2097152	254

- Netz ID bzw. Subnet ID sind Grundlage für Routing-Entscheidung
- schnelle Erkennung der Länge der Netz ID anhand Präfix
⇒ Performanzgewinn

Rechnernetze

Cidr:

- CIDR: Classless Interdomain Routing
- Struktur:

Network prefix	Subnet ID	Host ID
----------------	-----------	---------
- Notation: <network>/<prefix length>
- Beispiel: 192.168.121.0/26
die ersten 26 Bits bilden Netz ID; der Rest ist Host ID
- Zweck: Grenzen zwischen Netz ID und Host ID werden flexibel
- Bei CIDR müssen im Routingprotokoll die Präfixe in den Routenankündigungen übermittelt werden
- Adresszuweisungen müssen die Netztopologie berücksichtigen (hierarchisches Routing)
- RFC 1517 bis 1520
- Subnetting auch bei CIDR möglich durch Verwendung von Variable Length Subnet Mask

Quittungen:

Varianten von Quittungen

- Positive/negative Quittungen für **einzelne** PDUs
- Sammelquittung
 - einzelne Quittung für **mehrere aufeinander folgende** PDUs
→ geringere Anzahl Quittungen
- Selektive Quittung
 - für eine **Teilmenge** bisher gesendeter PDUs
 - ermöglicht selektive Wiederholung

Wiederholung

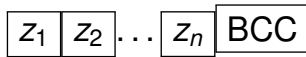
- Go-Back-N
 - wenn PDU mit snn=N verloren
 - Wiederholung aller PDUs ab snn=N
- Selektive Wiederholung
 - Wiederholung einzelner nicht quitierten PDUs
 - Leistungsvorteil bei großen Sequenznummernräumen (z.B. bei hohem RTD)

Rechnernetze

Fehlererkennung:

BCC:

- Nachricht besteht aus Folge von Zeichen aus m Bits
- Übertragen wird um Parität erweiterte Zeichenfolge und Prüfzeichen BCC



z_1	=	$b_{1,1}$	$b_{1,2}$	\dots	$b_{1,m}$	$b_{1,m+1} \leftarrow$ Querparität
z_2	=	$b_{2,1}$	$b_{2,2}$	\dots	$b_{2,m}$	$b_{2,m+1}$
\vdots						
z_n	=	$b_{n,1}$	$b_{n,2}$	\dots	$b_{n,m}$	$b_{n,m+1}$
BCC = z_{n+1}	=	$b_{n+1,1}$	$b_{n+1,2}$	\dots	$b_{n+1,m}$	$b_{n+1,m+1}$
		\uparrow				
		Längs-				
		parität				

- $b_{n+1,m+1}$ ist Spezialfall
- Art der Parität muss vereinbart werden

CRC:

- CRC: Cyclic Redundancy Check
 - Sender und Empfänger vereinbaren Generatorpolynom G vom Grad r
 - Bitfolgen werden als **Binärpolynome** aufgefasst mit Arithmetik XOR
 - Prüfsequenz wird mittels G berechnet, dann übertragen und geprüft
- Übertragungsfolge U ist Sendefolge S gefolgt von Prüfsequenz Q

$$U = \begin{array}{|c|c|} \hline \text{Sendefolge (Nutzdaten)} & \text{Prüfsequenz (} r \text{ Bits)} \\ \hline \end{array}$$

$$U = S \cdot 2^r \oplus Q$$

wobei Q der Divisionsrest von $x^r \cdot S / G$ ist.

Beispiele für vereinbarte Generatorpolynome

Bei HDLC: $G = x^{16} + x^{12} + x^5 + 1$

Bei Ethernet:

$$G = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

In der Praxis, oft: CRC als rückgekoppelte Schieberegisterschaltung in HW realisiert

Rechnernetze

Rechenbeispiel: Senderseite

- Sei $G = x^5 + x^4 + x^2 + 1$ ($\equiv 110101$)
- Sei $S = x^9 + x^5 + x^2 + 1$ ($\equiv 1000100101$)

Berechne: $x^5 \cdot S/G = Q = x + 1$; somit $U = 100010010100011$

Empfängerseite

Empfänger bildet U/G

- Falls Division $\neq 0$, dann Fehler
- Falls Division ohne Rest, dann fehlerfrei
 - oder Fehler ist Vielfaches von G , d.h. nicht erkennbar
- Fehlerhafte Sendung: $H = U + F$ mit F Fehlerpolynom
- Ziel: G so wählen, dass $F \neq 0$ für gängige Fehlermuster

Hamming:

Beispiel: $b = 4$ und $r = 3$, also $n = 7$

- Prüfbits auf Positionen 1,2 4, Informationsbits auf 3,5,6,7

P	P	I	P	I	I	I
---	---	---	---	---	---	---

- Prüfbit 1 sichert ab 1,3,5,7, Prüfbit gerade Parität
 - Prüfbit 2 sichert ab 2,3,6,7
 - Prüfbit 3 sichert ab 4,5,6,7
 - Empfänger hat Zähler mit Initialwert 0
 - Bildet Parität über zu sichernde Bits
 - Falls Fehler, wird Prüfbitposition auf Zähler addiert
 - Zähler enthält Bitposition des falschen Bits
-
- Verfahren ausdehnbar auf Mehrbitfehler
 - Bei Distanz $h = 2 \cdot k - 1$ oder $h = 2k$ auf k Bits korrigierbar

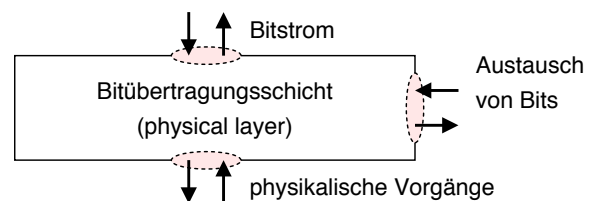
Rechnernetze

Schicht 1:

(Physical Layer/Bitübertragungsschicht)

Aufgaben und QoS:

- Transparenter Transport von Bits über data circuit
 - SDU = 1 Bit (seriell), n Bits (parallel)
 - Sequencing of bits
 - Kodierung von Bits in phys. Signale
- Protokoll legt Eigenschaften fest
 - physikalisch (Medien, Signale)
 - mechanisch (PIN-Gestaltung, Steckerkonfiguration)
 - funktional (PIN-Belegung, Takt)
 - prozedural (Ablauf der Elementarereignisse, Bedeutung)
- Dienstqualität der Schicht 1
 - error rate: Bitfehlerrate aufgrund Dämpfung, Rauschen, Dispersion. . .
 - Transit delay: f (Material, Länge)
 - Service availability: ist interessant bei geschalteten Ebene 1 – Verbindungen (X.21, ISDN-B-Kanal, SDH, ATM-SVC)
 - transmission rate: Funktion von
 - Spektrum: Frequenzbereich des Senders,
 - Bandbreite: Frequenzbereich des Mediums,
 - Codierung



Datenraten:

Rauschfreies Kanal (Nyquist): $C = 2 \cdot B \cdot \log_2 M$ [bit /s]

(C = Übertragungsrate, B = Bandbreite, M = Signalzustände)

Bsp. Telefon: $M = 2$, $B = 3100 \rightarrow C = 2 \cdot 3100 \cdot 1 = 6200$ bit/s

Bsp. AM-PSK: $M = 8$, $B = 3100 \rightarrow C = 2 \cdot 3100 \cdot 3 = 18600$ bit/s

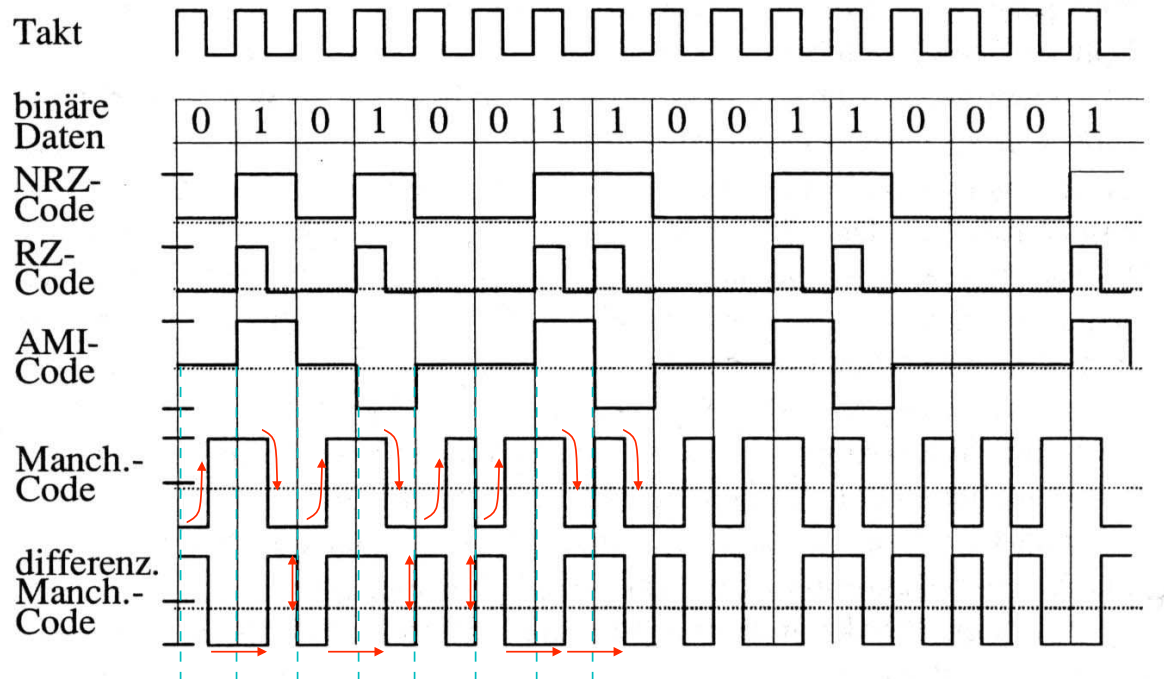
Kanal mit Rauschen (Shannon): $C = B \cdot \log_2 (1 + S/N)$ [bit /s]

(C = Übertragungsrate, B = Bandbreite, S/N = Rauschabstand = Singnalenergie S / Rauschenergie N, oft in dB: $y[\text{in dB}] = 10 \cdot \log_{10}(S/N)$)

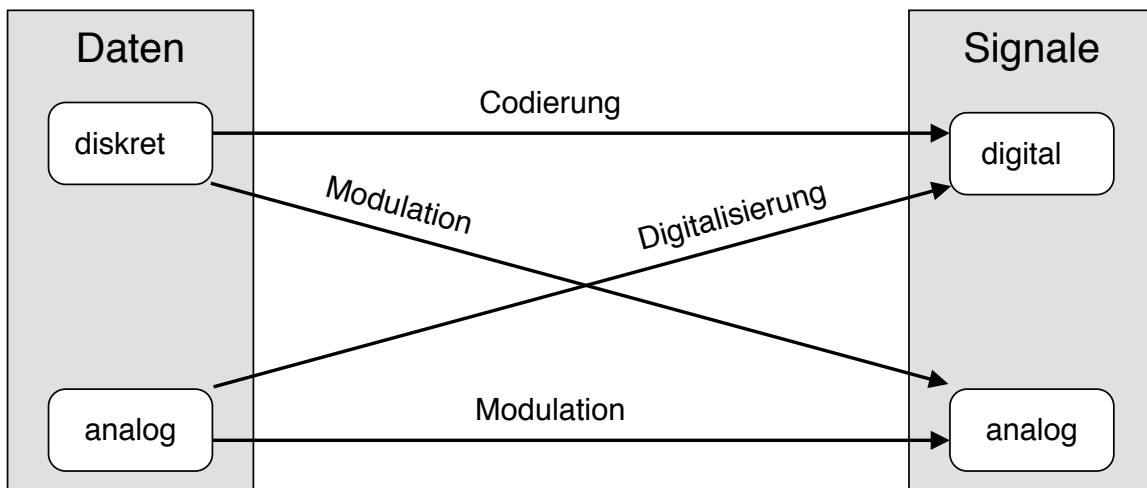
Bsp. Telefon: $S/N = 20\text{dB} \rightarrow 3100 \cdot \log_2 (1 + 100)$

Rechnernetze

Codierungen:



Datensignale



Digitalisierung

- Diskretisierung (Abtasttheorem betrifft Zeitachse)
- Quantisierung (betrifft Zerlegung der Wertachse)
- Codierung (Darstellung der Wertemenge)

Rechnernetze

Schicht 2: (Data-Link Layer/Sicherungsschicht)

Aufgaben und QoS:

- Framing
 - Zusammenfassung von Bits zu Blöcken/Frames (Rahmen)
 - Block/Frame-Synchronisation
- Fehlererkennung, ggf. Fehlerkorrektur
- Aufbau und Steuerung von Layer-2-Verbindungen
- Umgang mit dem Schicht 1 Data Circuit
 - Verschattung Medien- und Übertragungstechnik
 - Aus Data Circuits der Ebene 1 wird gesicherter Data Link / Logical Link auf Ebene 2, der medien- und übertragungstechnikunabhängig ist.

Protokollbezogen

- Bereitstellen und Steuern Layer-2-Connection (Data Link, Logical Link)
- Steuerung und Sichern Datenaustausch von
 - Zeichenfolgen (Blöcke) oder
 - Bitfolgen (Rahmen, *Frames*)

Dienstbezogen

- nach unten: Übernahme/-gabe von Bitfolgen
- nach oben: deckt Charakteristika der phys. Medien zu

Dienstgüte, Quality of Service

- Dauer Verbindungsaufbau, Durchsatz, Verzögerungen, Robustheit
- Schutz, Priorisierung
- Wahrscheinlichkeiten für...
 - Fehlschlagen des Verb.Aufbaus
 - Restfehler, Übertragungsfehler

Vielfachzugriffsverfahren:

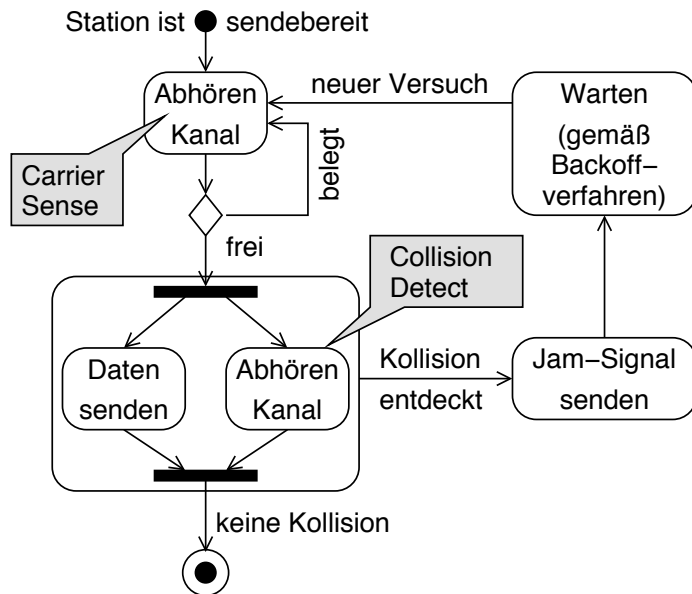
Aloha (Pure, Slotted, Reservation)

MACA

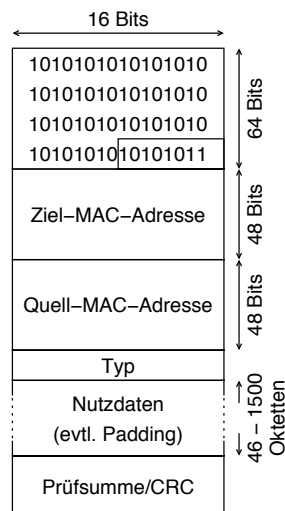
TOKEN RINGS

Rechnernetze

CSMA: (CD/CA)



- Präambel
 - 7 Bytes mit 10101010
- Start of Frame
 - 1 Byte 10101011
- Ziel-, Quell-MAC-Adresse
 - 48 Bits, flach
- Länge Datenfeld
 - Ethernetframe muss min. 64 Bytes lang sein (ab/mit MAC-Felder)
 - Nutzdaten (0–1500 Bytes)
 - Pad (0–46 Bytes bedarfsweise)
- Prüfsumme
 - 4 Bytes (CRC-Verfahren)



Protokolle (BSC, HDLC, PPP, VLANs, ATM):

BSC (Zeichenorientiert):

- BSC (Binary Synchronous Communication)
 - IBM, 70er Jahre
 - Beispiel für Basic Mode Procedures (DIN 660019)
- Verwendung nur noch bei "dummen" Terminals (z.B. 327x), Geldautomaten
- Codeabhängig (ASCII oder EBCDIC)
- Prüfsummenbildung mit **BCC**
- Bit- und Bytesynchronisation durch spezielle **Steuerzeichen**
- Sehr viele Steuerzeichen → Transparenz über **Fluchtsymbol**

Rechnernetze

SOH Start of Header, STX Start of Text, ETX End of Text, ETB End of Transmission Block, SYN Blocksynchronisation, EOT End of Transmission, PAD Füllzeichen, ENQ Polling, DLE Fluchtsymbol, ACK positive Quittung, NAK negative Quittung

Nutzdaten:

A	B	STX	C	DLE	D
---	---	-----	---	-----	---

Transparenz durch Fluchtsymbol

Übertragener Block:

SYN	SYN	STX	A	B	DLE	STX	C	DLE	DLE	D	ETX
-----	-----	-----	---	---	-----	-----	---	-----	-----	---	-----

Blockbildung mit Fluchtsymbol

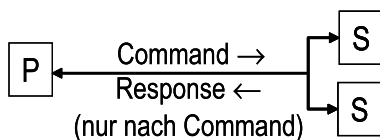
Übertragener Block:

SYN	SYN						
DLE	STX						
(Blockbeginn)							
A	B	DLE	STX	C	DLE	DLE	D
(Info)							
DLE	ETX						
(Blockende)							

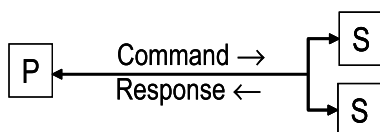
DLE : Byte-Stuffing

HDLC:

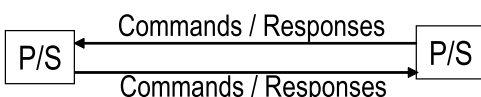
- HDLC (High Level Data Link Control)
- Protokollklasse **bitorientierter** Prozeduren der Schicht 2
 - 3 Basisklassen
 - 13 Optionen
- bekannte Varianten sind
 - LAP B (X.25)
 - LLC1/LLC2 bei Ethernet/Token Ring (IEEE 802.2)
 - LAP D bei ISDN D-Kanal
- Definiert in ISO 3309, 4335, 6159, 6256, 7809, 8886
- Unterstützt **Punkt-zu-Punkt** sowie **Mehrpunkt**, alle Betriebsarten
- Arbeitet **synchron**
- Absicherung mittels **CRC, Sequenznummern, Fenstertechnik**



Normal Response Mode

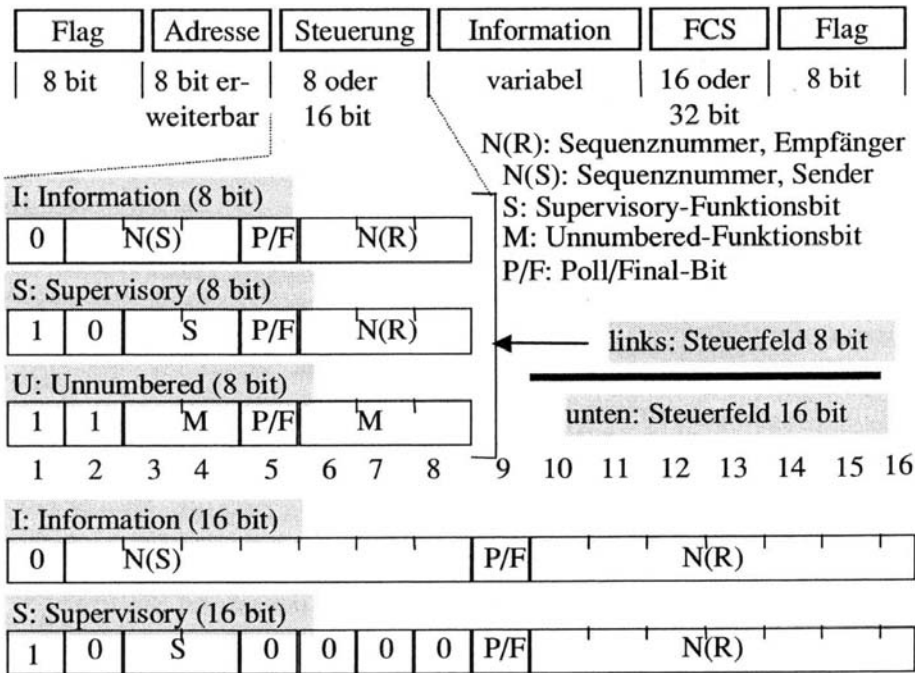


Asynchronous Response Mode



Asynchronous Balanced Mode

Rechnernetze



Erweitertes Adressfeld: n Felder mit je 8 bit, das erste Bit der Felder 1, ..., $n - 1$ enthält 0, das erste Bit des Feldes n enthält 1

- Flag: 01111110, Transparenz über Bitstuffing
- Adressen nur bedeutend bei Mehrpunktverbindungen
- Normalfenster ist 8; erweiterbar auf 128
- Normal CRC-16 ($x^{16} + x^{12} + x^5 + 1$), bei LLC CRC-32
- Poll/Final:
 - Poll (Abfrage) durch P
 - S deutet Ende der Sendung mit F (Final) an
 - Mit P/F kann Checkpoint erzwungen werden.
- Systemparameter (d.h. Variable der Protokollimplementierung)
 - Zeitüberwachung für Quittung auf I und für F-Bit und in DTE
 - Wiederholungszähler, Framelänge, aktuelles Fenster

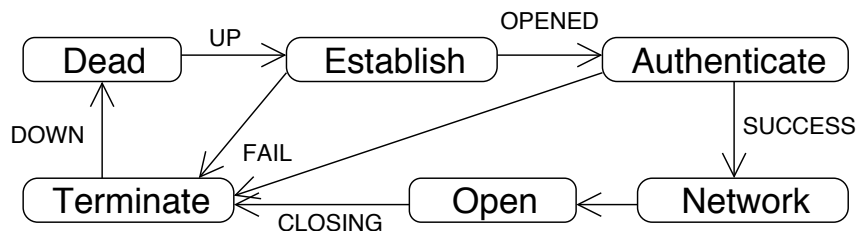
Rechnernetze

PPP:

- Point-to-Point Protocol (PPP)
 - definiert in RFC 1661 (RFC 1662, 1663)
 - Ziel: einfach zu konfigurieren
 - Häufiger Anwendungsfall: Wählleitungen (Modem, ISDN, DSL)
 - Rahmenaufbau ähnlich zu dem von HDLC
 - Zeichenorientiert, überträgt Vielfache von 8 Bits
 - Nutzung über HDLC, SDH/SONET, Ethernet, ATM
- LCP (Link Control Protocol)
 - in PPP-Rahmen transportiert
 - Auf-/ Abbau des Links
 - Aushandlung von Optionen
- NCP (Network Control Protocol)
 - Bezug zum (transportierten) Protokoll der Vermittlungsschicht
 - Familie von Protokollen (verschieden, je nach Ausprägung der Schicht 3)

A link control protocol for bringing lines up, testing them, negotiating options, and bringing them down again gracefully when they are no longer needed. This protocol is called **LCP (Link Control Protocol)**. It supports synchronous and asynchronous circuits and byte-oriented and bit-oriented encodings.

A way to negotiate network-layer options in a way that is independent of the network layer protocol to be used. The method chosen is to have a different **NCP (Network Control Protocol)** for each network layer supported.



- Zustand *Establish*: Aushandlung von Übertragungsoptionen mit LCP
- Zustand *Network*: Aufruf des NCP (z.B. Zuweisung von IP-Adressen)

Bytes	1	1	1	1, 2	...	2, 4	1
	Flag	Address	Control	Protocol	Payload	CRC	Flag

- Flag: Zeigt Beginn/Ende des Rahmens an (01111110)
- Address: Schicht-2-Adresse (default: 0xFF = Broadcast)
- Control: z.B. Fenstergröße (kaum benutzt, default 00000011 = *Unnumbered frame*)

Rechnernetze

Virtuelle VLANs:

- VLAN-bildung über **Ports** der Koppelkomponente (z.B. Switch-Ports)
 - Anforderung: alle DEE hinter einem Port gehören VLAN an
 - Einfache Konfiguration: Port-Bereiche werden VLAN fest zugeordnet
 - Mobilität der DEE problematisch: Portwechsel → VLAN-Wechsel
- VLAN-bildung über **MAC-Adressen** der DEE
 - Anbindung einer DEE unabhängig von Port (MAC-Adresse bleibt gleich)
 - MAC-Adressen neuer DEE müssen VLAN zugeordnet werden
→ Konfig.-Aufwand
- VLAN-bildung über **Adressen der Vermittlungsschicht**
 - Verletzung des Schichtenkonzeptes!
 - keine Transparenz bezüglich Schicht-3-Protokollen (z.B. versch. Versionen des Internet Protocol)

ATM:

- ATM = Asynchronous Transfer Mode
 - Rein *verbindungsorientiert* auf Schicht 2/3
 - Multiplex von Zellen (= Pakete fester Länge)
 - Zelle: 5 Byte Header, 48 Byte Nutzdaten
 - Koppelkomp.: Switches, Cross-Bars
 - Garantien bezüglich der Dienstgüte (QoS)
- Diensttypen
 - PVC (Permanent Virtual Circuit)
 - Fest geschaltete Verbindung zwischen Endpunkten
 - z.B. (virtuelle) Standleitung
 - SVC (Switched Virtual Circuit)
 - Dynamisch eingerichtete Verbindung
 - z.B. garantierte 64 kbit/s-Kanal für ein Telefonat

Rechnernetze

Schicht 3: (Network Layer / Vermittlungsschicht)

Aufgaben:

- Pfadschaltung (Vermittlung) zwischen zwei Endsystemen unter Berücksichtigung von Transitsystemen und Transitnetzen auf Basis eines netzglobalen Adressraumes
- Wegewahl (Routing)
- Dienstgüte verhandeln
- Bereitstellung eines verbindungslosen oder verbindungsorientierten Netzdienstes

Hinweis: Wegewahl und Vermittlung können auch technologieabhängig auf anderen Schichten vorkommen, so z.B. auf Ebene 7 (E-Mail-Relays), Ebene 2a (Bridges im MAC-Layer), Zellvermittlung bei ATM

QoS:

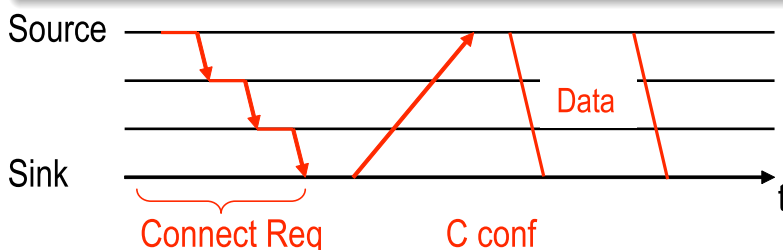
- Verbindungsaufbauwahrscheinlichkeit (Blockierwahrscheinlichkeit)
- Verbindungsaufbauzeit
- Durchsatz der Schicht-3-Verbindung
- Nachrichtenübertragungszeit
- Schwankung in der Übertragungszeit (Jitter)
- Restfehlerrate

Vermittlungsverfahren:

Leitungsvermittlung:

Leitungsvermittlung, Durchschaltung, Circuit Switch

- Vor Übertragung Aufbau des Pfades von Sender zu Empfänger
- Dedizierter Kanal für die gesamte Datensphase
- typisch ist gleicher Grenzdurchsatz für die Links
- typisch gekoppelt mit VO-Dienst
- Beispiel: POTS, ISDN-B-Kanäle

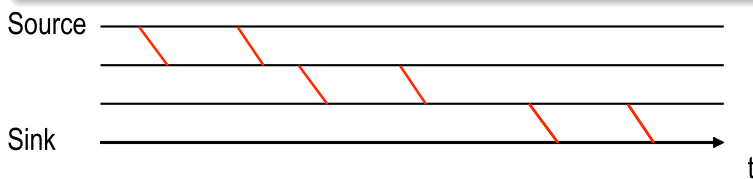


Rechnernetze

Nachrichtenvermittlung:

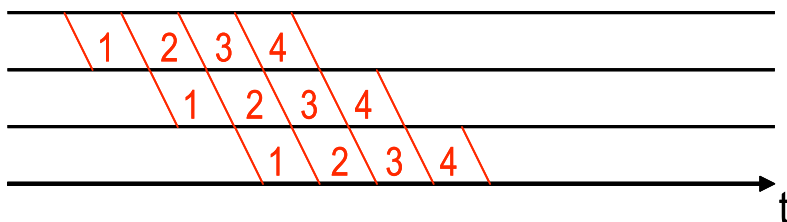
Nachrichtenvermittlung, Message Switching, Store and forward, Speichervermittlung

- Nachricht wird als Ganzes versendet
- Jeder Knoten auf Weg speichert Nachricht und sendet dann auf nächstes Streckenstück
- Streckenstücke sind i.a. nicht homogen
- Schwankendes Processing und Queuing Delay
- Beispiel: Email, IP



Paketvermittlung:

- Zerlegung der Nachrichten in etwa gleichlange Pakete.
- Senden der Nachrichten paketweise nach Prinzip Message Switching
- ⇒ Eine Art "Pipelining Effekt", bessere Leitungsauslastung
- ⇒ Reihenfolgeproblem
- VO und VL möglich
- Beispiel: X.25, ATM - Zellvermittlung



Rechnernetze

Routing:

Begriffe

- Routingalgorithmen beschreiben Wegewahlverfahren
- Verfahrensauswahl und -ausprägung hängt ab von der Routingstrategie (Policy), also von den *Zielfunktionen*(„Kostenfunktionen“), z.B.
 - geringe Ü-Kosten,
 - geringe Ü-Zeiten,
 - gute Leitungsauslastung,
 - großer Durchsatz
- Verfahren soll sein
 - einfach (Alg.-Komplexität, Netzoverhead),
 - adaptiv (Last, Topologie),
 - robust (bei Fehlern), fair
- Grundlage ist Routing-Tabelle

Probleme

- Zielkonflikte
- Beschreibung der Topologie
 - wie beschrieben (Leitungen, Knoten, Kosten)
 - vollständig / partiell
- Berechnung
 - wo (zentral / dezentral)
 - welche Info wird vom Algorithmus benötigt
 - wie wird Info bereitgestellt
 - welche Ereignisse stoßen Berechnung an
 - wann wird neuer Weg aktiviert

Routing Tabelle:

- realisiert Vorgabe zur Weitergabe von Nachrichten
- bestimmt die „Entscheidungsfindung“ in IWUs/Routern
- Default Route: spezieller Eintrag; wird benutzt, wenn kein anderer (spezifischerer) Eintrag passt.

Felder eines Tabelleneintrags (eine Zeile):

- Ziel (Netz oder Host)
- Gateway (d.h. Router)
- Netzmaske (Angabe zur Bestimmung der Netzadresse)
- Metrik (Kostenwert)
- Schnittstelle/Iface (Zielleitung)

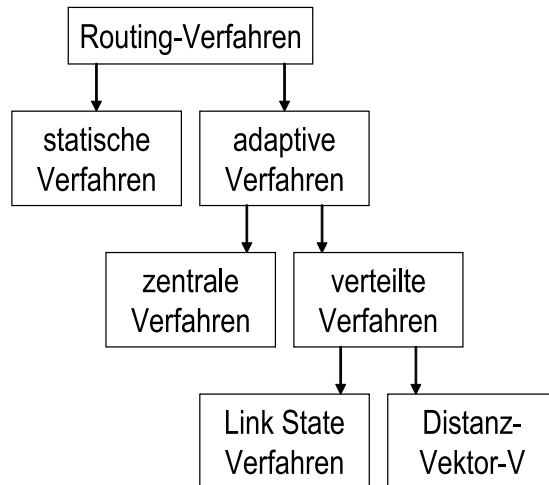
Auf Unix-Derivaten (z.B. Linux): `$ route -n`

Rechnernetze

Algorithmen:

Grundidee: Optimaler QSB (Quelle-Senke-Baum) aufbauen.

Verschiedene Verfahren:



- Weitere Verfahren
 - isoliertes Verfahren, nicht adaptiv: Flooding
 - isoliertes Verfahren, lastabhängig: Hot Potato

Static Routing:

Routingtabelle wird *einmal* aufgrund festgelegter Routing-Metriken erstellt

Ziel z	Nachbarknoten auf dem Weg $j \rightarrow z$					
	1. Wahl		2. Wahl		3. Wahl	
	Nr.	Gewicht	Nr.	Gewicht	Nr.	Gewicht
	1	0,63	6	0,21	4	0,16

- Berechnen Zufallszahl x aus $[0,00 \text{ bis } 0,99]$
 - $x \leq 0,63 \rightarrow$ nimm 1. Wahl
 - $0,63 < x \leq 0,84 \rightarrow$ nimm 2. Wahl
 - $x > 0,84 \rightarrow$ nimm 3. Wahl
- Grenzfälle:
 - nur eine feste Wahl
 - Alternativen nur bei defekter 1. Wahl (backup trunk)
- Wertung: Verfahren einfach, jedoch nicht adaptiv

Hot Potato:

- isoliertes Verfahren
- ankommendes Paket wird als hot potato betrachtet und auf abgehende Leitung mit kürzester Warteschlange gelegt
- Evtl. erhebliche Umwege, falls gewählte Leitung nicht optimal zum Ziel führt
- empfindlich gegen Überlast

Rechnernetze

Flooding:

- Idee: Jede eintreffende Nachricht wird an alle Nachbarknoten geschickt
- Problem: es entstehen beliebig oft Kopien
- Maßnahmen:
 - nicht zurück an sendenden Knoten
 - Lebensdauerbegrenzung durch Timer oder Hop Count
 - Nachrichtenmerkmal speichern
- Wertung:
 - Verfahren erzeugt Zusatzlast, aber sehr einfach
 - Verfahren extrem robust (z.B. Milit.Netze)
 - kann benutzt werden für schnelle multiple update von DB
 - enthält immer optimalen Weg (Metrik gegen andere Verfahren)

Adaptives Verfahren mit Nachbarkenntnis (Distanz-Vektor):

- Beispiel: Distanz-Vektor-Alg. (Bellman-Ford-Alg)
- wird im Routing Information Protocol (RIP) im Internet benutzt
- Merkmale:
 - jeder Knoten hat Routingtabelle (Adresse, Distanz),
 - Update geschieht periodisch
 - jede Kante ist mit Gewicht belegt
 - Distanz ist Summe der Gewichte auf dem zum Ziel
- Initialisierung mit Eintrag, . . .
 - dessen Ziel dem lokalen Knoten entspricht,
 - dessen Next-Hop nicht angegeben ist und
 - dessen Distanz auf 0 gesetzt ist

Wertung

- Verfahren sehr einfach, Rechenaufwand gering
- Nur partielle Topologie muss bekannt sein
- Zusätzliche Netzbelastung
- Wegewahlinfo breitet sich nur langsam aus, kann zu Inkonsistenzen führen!

Gemeinsame adaptive Wegberechnung (Link-State Routing, SPF Routing):

Ziel

- möglichst alle Knoten sollen aktuelle Wegewahlinfo haben.
- langsame Ausbreitung von Anpassereignissen soll vermieden werden

Verfahren

- Basis ist SDF-Algorithmus
- jeder Knoten kennt Topologie und Bewertung
- jeder Knoten enthält von jedem anderen Knoten j dessen lokalen Verzögerungsvektor $[k, D_{jk}]$ für alle $k \in A(j)$
- Verteilung Vektoren z.B. durch Flooding

Rechnernetze

Link-State-Routing, SPF-Routing

- wird im Internet im Routingverfahren OSPF verwendet
- jeder Router versucht seine Nachbarn kennenzulernen
- jeder Router bildet ein Link State Packet (LSP) mit Namen der Nachbarn und Gewichten der zugehörigen Links
- LSP werden an alle Rechner geschickt, jeder Rechner speichert die zuletzt erhaltenen LSP aller anderen Router. Damit kennt jeder Router die vollständige Netztopologie
- Berechne QSB nach SDF-Verfahren (Dijkstra)

Autonome Systeme:

Autonomes System (AS)

- Eine Menge von Routern unter der Kontrolle *einer* administrativen Domäne (AS ist *Routing Domain*)
- AS sind genehmigungspflichtig (durch IANA/ICANN)
- Nutzung eines **Interior Gateway Protocol** (IGP) im Inneren des AS
- Nutzung eines **Exterior Gateway Protocol** (EGP) zwischen AS
- AS hat eindeutige numerische ID (z.B. LRZ/MWN: AS 12816)
- Abkommen zwischen AS (ggf. mit Zahlungen verbunden)
 - **Peering**: Routen **in/aus** Netz eines fremden AS
 - **Transit**: Routen **durch/über** das Netz eines fremden AS
 - Technische Realisierung durch **Network Exchange**
- IGP (intra-AS)
 - RIP (Routing Information Protocol): Distanz-Vektor-Algorithmus
 - OSPF (Open Shortest Path First): Link-State-Routing
- EGP (inter-AS)
 - BGP (Border Gateway Protocol): ähnlich Distanz-Vektor-Verfahren

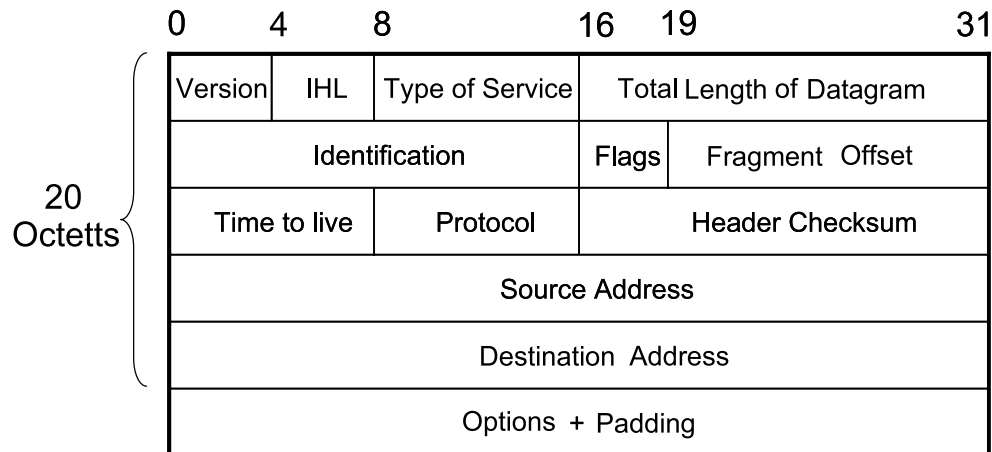
IP

- Verbindungsloser Dienst, zwei Dienstprimitive SEND, DELIVER
- Parameter:
 - Source address, destination address: IP-Adresse Sender-/ Empfänger-Host
 - Protocol: ID des IP users (SAP-Adresse), z.B. 1 ICMP, 2 IGMP, 6 TCP, 8 EGP, 9 UDP, 46 RSVP, 89 OSPF
 - type of service (ToS) indicator
 - Priorität (8 Stufen)
 - Reliability (2 Stufen)
 - Delay (normal/low)
 - Thruput (normal/high)
 - identifier¹: eindeutige Kennung für Reassembly und Error Reporting
 - "don't fragment" id¹
 - time to live (TTL)¹, gemessen in *hops*
 - data length (in Bytes)
 - option length, option data, data

Rechnernetze

IP-Header:

Das IP-Protokoll ist datagramm-orientiert



- Version: Version des IP-Protocols, derzeit 4
- IHL: Internet Header Length in 32-Bit-Worten (Minimum= 5), d.h. kleinster Header = 20 Oktetten
- Length: in Bytes
- Flags:
 - M "More bit" (für Fragmentierung und Wiederherstellung (*Reassembly*)
 - M = 0 folgt nichts (letztes Teilpaket)
 - M = 1 (folgen weitere Teilpakete)
 - DF "Don't fragment bit"
- Fragment Offset: benutzt für Reassembly, zählt in Anzahl von 64-bit-Einheiten für den Datenanteil in Bezug auf Datenanfang
- Padding: Auffüllen des Headers auf Vielfaches von 32-bit-Einheiten
- Data:
 - variabel lang
 - Vielfaches von 8 bit
 - max. Datagramm-Länge (inkl. Header): 64KByte = 65535 Oktetts

Rechnernetze

IP Unterstützende Protokolle:

- ICMP (Internet Control Message Protocol), RFC 792
 - Benachrichtigungsprotokoll, wirkt unterstützend für Fehler- und Mgmtbehandlung, nötige IP-VL-Dienst
 - Destination unreachable, time exceeded, Parameterproblem (Syntaxfehler im IP-Header), source quench (bei Router- Überlast), Redirect (Routing-Korrektur), Echo, Timestamp, usw.
 - Basis für traceroute
- DHCP (Dynamic Host Configuration Protocol)
Dynamische temporäre Zuweisung von IP-Adressen
- DNS (Domain Name Service)
 - Abbildung von Namen auf IP-Adressen
 - Interaktion zwischen Resolver (Client) und Server (Directory)
- ARP (Address Resolution Protocol), RFC 826
Abbildung von IP-Adressen auf MAC-Adressen

IP v6

Motivation

- Adressknappheit bei IPv4 bei wachsendem Bedarf an Adressen
- Neue Anforderungen: QoS, Mobilität ...

Neue und ausgebaute Funktionalität im Vergleich zu IPv4

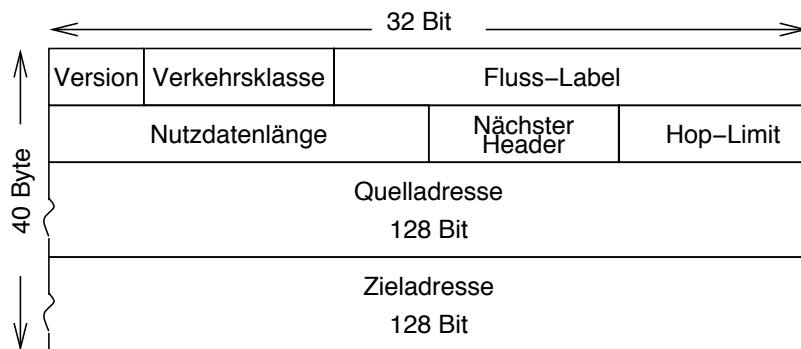
- erweiterter Adressraum (128 statt 32 bit)
- Erweiterungen für QoS auch bzgl. Streams wie Sprache, Video
- dynamische Adresszuweisung (Mobilitätsunterstützung)
- Ressource Allocation (Reservierungsprotokolle)
- Security capabilities (privacy, authentication)
- hop-by-hop-options (Routeranweisungen)

Status

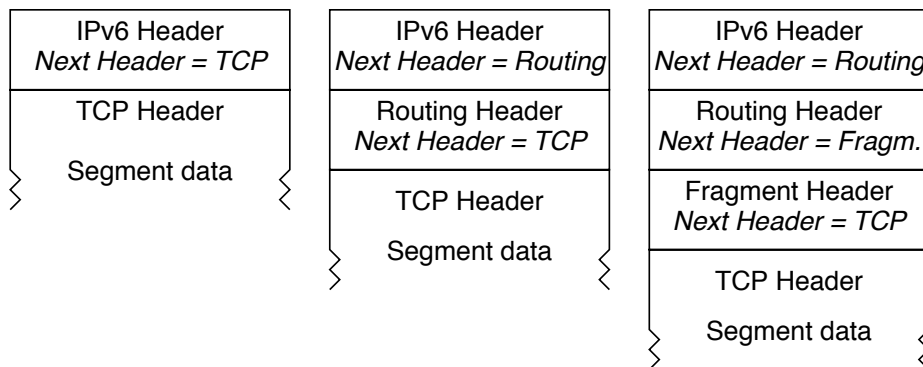
- IPv6 wächst; Implementierungen in gängigen BS umgesetzt
- Vorreiter China, Japan. Grund u.a. mangel an IPv4-Adressblöcken

Rechnernetze

IPv6 Header:



- Feste Headerlänge von 40 Byte → Header-Längenfeld entfällt
- Keine Header-Prüfsumme → keine Neuberechnung bei jedem Hop
- *Verkehrsklasse* entspricht Type-of-Service Feld bei IPv4
- *Fluss-Label* Zuordnung von Paketströmen/-flüssen
- *Nächster Header* entspricht Protokollfeld in IPv4-Header
kann alternativ auf *Extension Header* verweisen
- *Hop-Limit* entspricht TTL-Feld bei IPv4



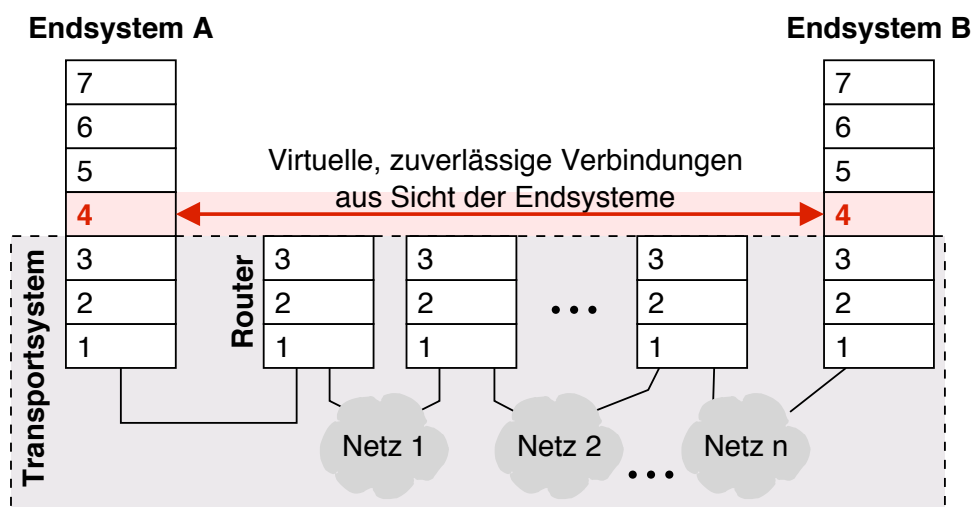
- “*Nächster-Header*”-Feld gibt Typ des folgenden Headers an
- Anzahl (i.d.R. eins) und Reihenfolge der Extension Header vorgegeben → effiziente Abarbeitung
- IPv6-Spezifikation (RFC 2460) fordert mindestens:
 - Hop-by-Hop Options, Routing, Fragment, Destination Options, Authentication, Encapsulating Security Payload
- Zusätzliche Header-Typen möglich, z.B. für Mobilitätsunterstützung

Rechnernetze

Schicht 4: (Transport Layer/ Transportschicht)

Aufgaben und Überblick

- Netzunabhängiger Transport von Nachrichten zwischen Endsystemen
- Verschattung der Wege durchs Netz
- Fehlerbehandlung Ende-zu-Ende
- Anpassung der Übertragungsqualitäten, Netzauswahl
- Verbindungsloser oder -orientierter Dienst
- z.B. im Internet: TCP (verbindungsorientiert), UDP (verbindungslos)
- Splitting/Multiplex über Anwendungen/Prozesse
- Verbindungen bestehen zwischen zwei Endsystemen
 - Transitnetze bzw. Netzknoten für Transportprotokoll nicht sichtbar
 - ⇒ Pfad unbekannt (bzw. irrelevant)
- Virtuelle Verbindung



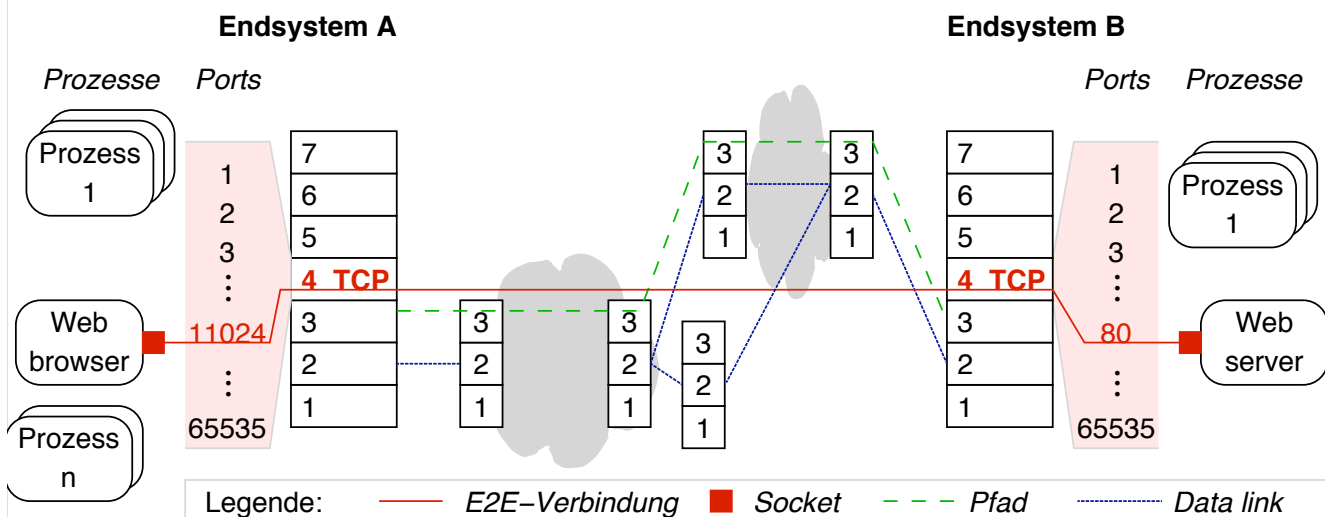
Rechnernetze

TCP Transmission Control Protocol

- Internet Transportprotokoll, RFC 793
- TCP unterstützt Ende-zu-Ende-Transportverbindungen (Point-to-Point)
 - voll duplex,
 - mit Fehlerbehandlung
 - mit Flusssteuerung (Überlastkontrolle)
- TCP ist **Byte-Strom-orientiert**, Sequenz- und Quittungsnr. beziehen sich auf Bytes
- TCP bietet **VO-Dienst**, Aufbau mit **3-way-handshake**
- TCP-Nutzer sind über **Sockets** adressierbar
 - Socket: (IP-Adresse Host, lokale PortNr)
 - TCP-Verbindung=(socket1, socket2)
- TCP unterstützt ein Multiplexen von Anwendungen
- TCP kann Dienstdaten zwischenspeichern, tatsächliche Transport-Blockung kann ein TCP-Nutzer nicht sehen

TCP Socket

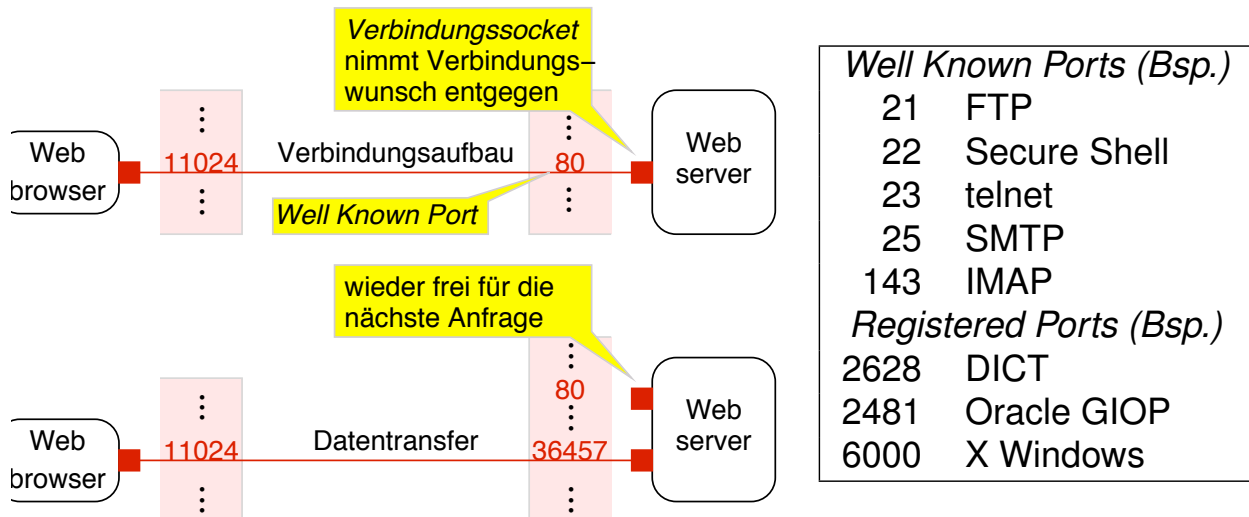
- Ende-zu-Ende-Verbindung an Dienst bzw. Applikation gebunden
- **Socket**: Endpunkt für Kommunikation
 - beschrieben durch Schicht-3-Adresse und **Portnummer**
 - wird erzeugt von und „gehört“ einem Prozess
 - bietet Schnittstelle für zuverlässige Byte-Übertragung
- Verbindung gegeben durch ein Socket-Paar



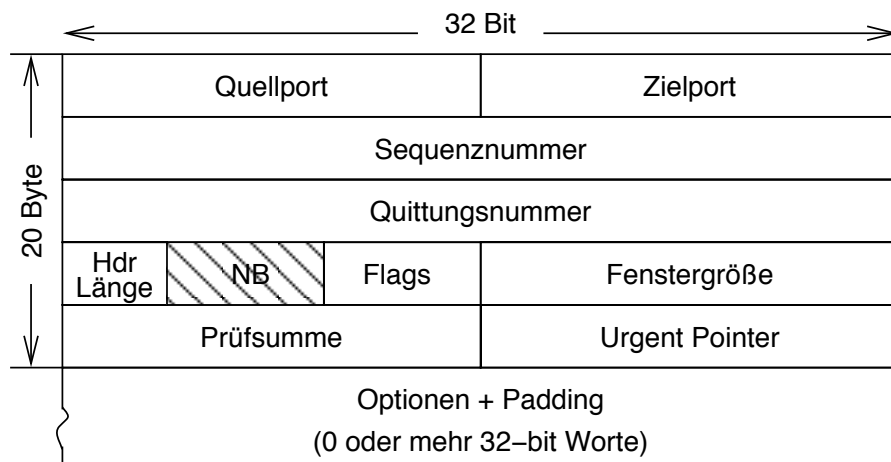
Rechnernetze

Ports

- Portnummern vergibt IANA (Internet Assigned Numbers Authority)
 - Well Known Ports* 0 – 1023: 1-255 für TCP-Anwendungen, 256-1023 für Unix-Anwendungen
 - Registered Ports* 1024 – 49151: meist herstellerspezifisch
 - Dynamic and/or Private Ports* 49152 – 65535: frei nutzbar
 - Unix-Systeme: Abbildung Port/Dienst in Datei `/etc/services`



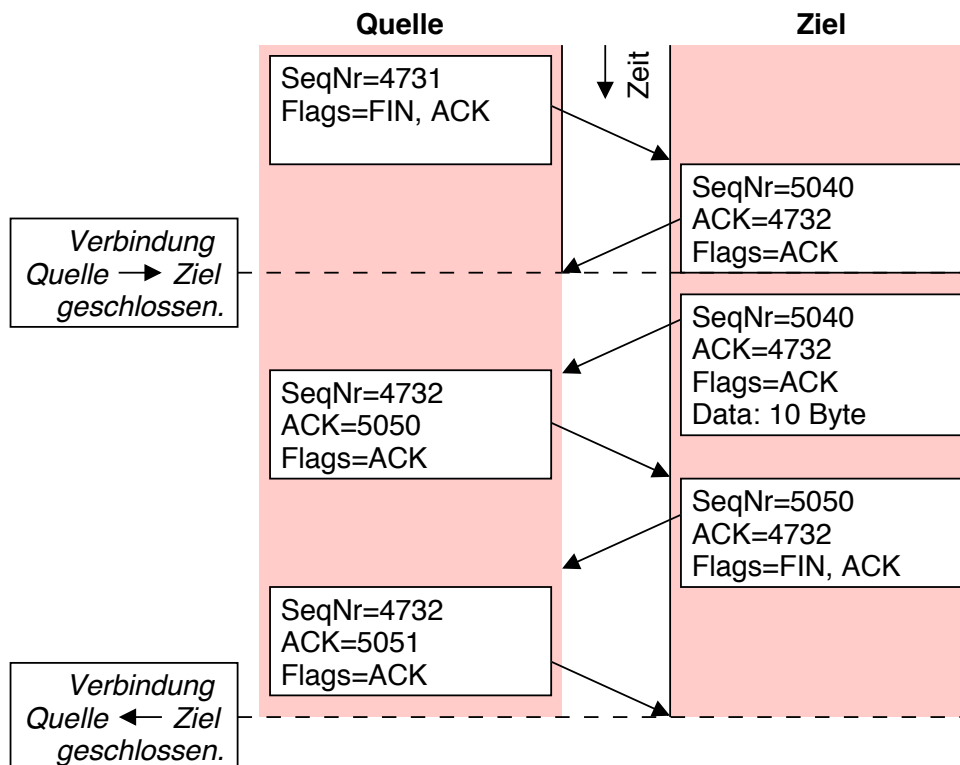
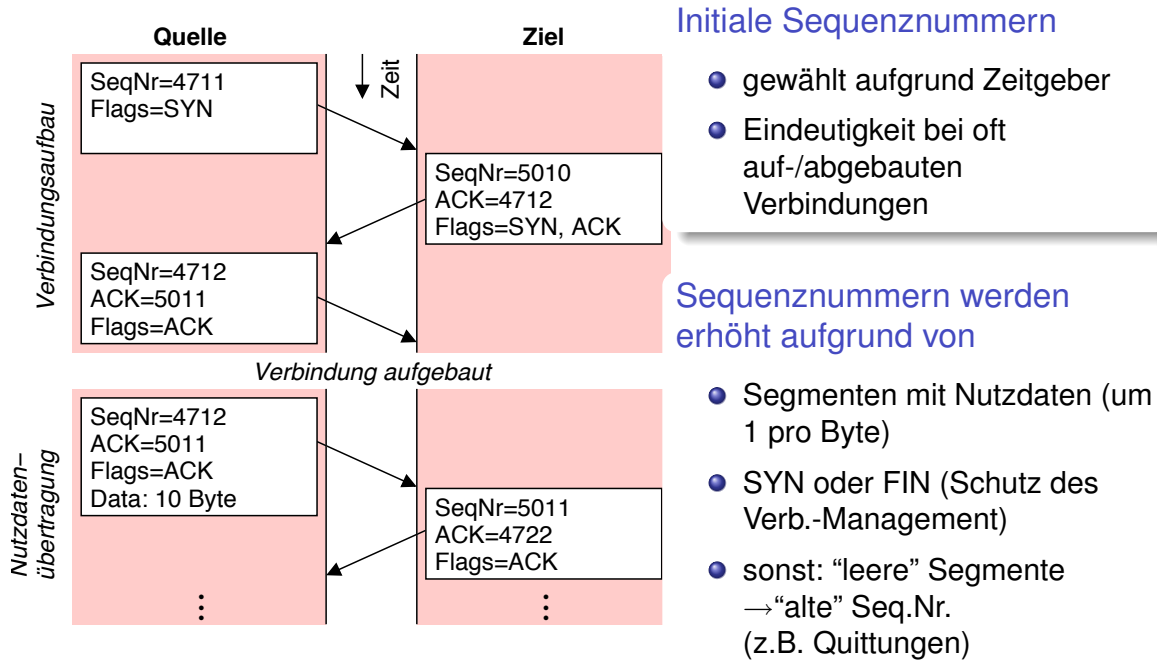
TCP Header



- Eine TCP-PDU wird oft auch als **Segment** bezeichnet
- Source Port, Destination Port → Anwendungsprozesse
- Sequenznummer
 - Senden und Empfangen (zählt auf Byte-Strom)
 - Sequenznr. eines Segments ist Bytestromnr. des ersten Bytes im Segment
- Header-Länge (4 Bit) in Anzahl 32 Bit-Wörter
- NB: nicht benutzt, 6 Bits (schraffiert)

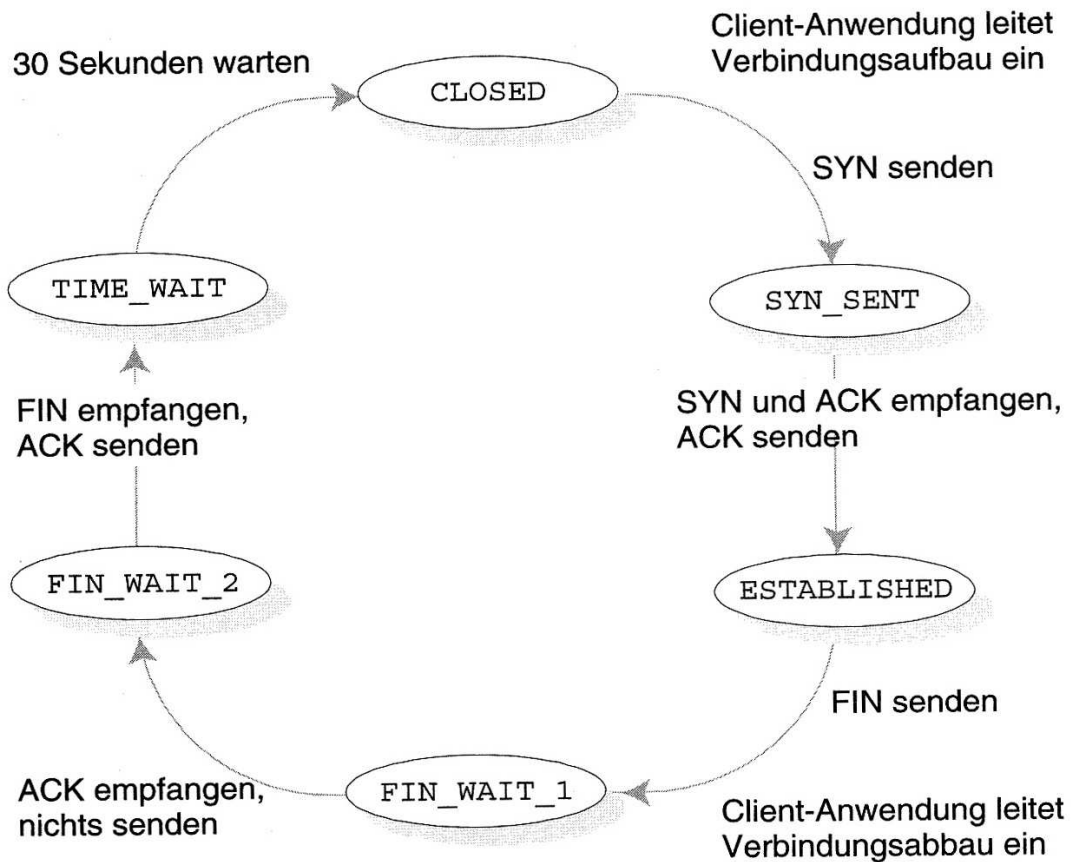
Rechnernetze

3-Way-Handshake

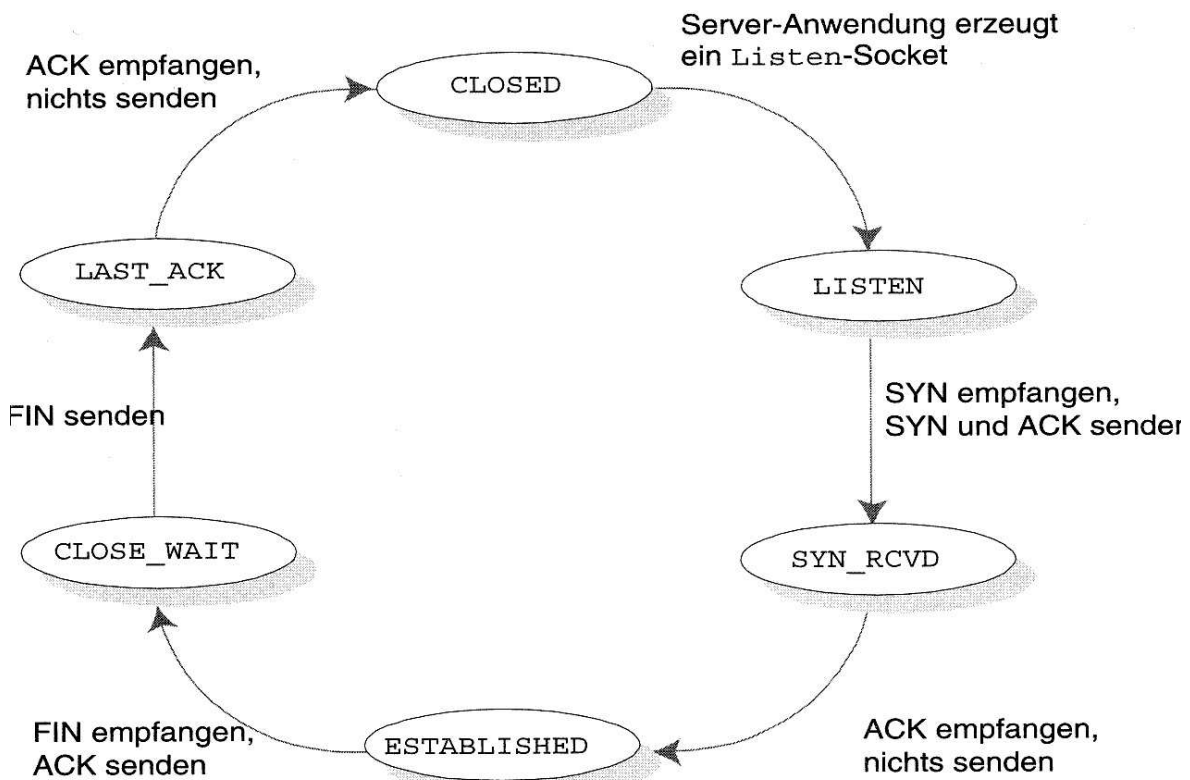


Rechnernetze

Zustände Client



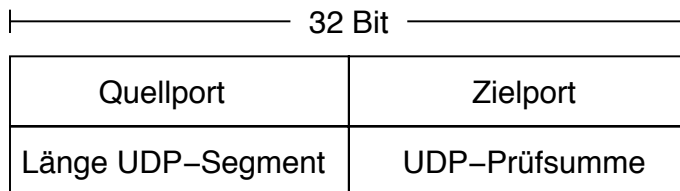
Zustände Server



Rechnernetze

UDP User Datagram Protocol

- verbindungsloses, unzuverlässiges Transportprotokoll
 - (kein Verbindungsaufbau, kein Verb.-Status, unregulierte Senderate)
- Multiplex von Anwendungen mittels Ports (wie TCP)
- TFTP, DNS, RPC, SNMP werden z.B. über UDP abgewickelt
- Header festgelegt in RFC 768
 - (nur 8 Byte im Gegensatz zu 20 Byte bei TCP)



- Programmieren mit UDP-Sockets
 - Senden/empfangen isolierter Datagramme
 - nur Sicht auf lokales Socket
 - Senden/empfangen zu/von mehreren Hosts
 - IP-Adresse und Port des Empfängers müssen i.d.R. beim Senden eines jeden Datagramms angegeben werden
 - Reihenfolgesicherung, Behandlung von Verlust, Steuerung Senderate: obliegen Anwendungsprogramm

Rechnernetze

Middleware

Motivation

Probleme bei im Netz verteilte Anwendungen (verschieden Darstellung von Daten, Systemschnittstellen, Programmiersprache der Komponenten, ecc)

Lösung

- Einführung einer **Middleware**, die Teile einer Anwendung verbindet
- Verschattung des Zugriffs auf entfernte Komponenten
- Spezifika von Programmiersprachen im Vordergrund, statt denen der Anwendung selbst

Prozeduraufrufe Basisfunktion.

- Aufruf einer Funktion/Methode auf entfernter Komponente.
- Übergabe von Argumenten; Rückgabewerte

Typensystem ⇒ Homogenisierung der Datenformate

- Gleiche Typen aus Sicht der Komponenten
- Transparente Umwandlung zwischen Systemen mit verschiedenen Formaten

Überblick

Beispiele für Kommunikations-Middleware

- Unix RPC
- CORBA^a (Common Object Request Broker Architecture)
- Java Remote Method Invocation (Java RMI^b)
- SOAP^c (Simple Object Access Protocol)

^aOMG, Object Management Group

^bSun Microsystems

^cW3C, WWW Consortium

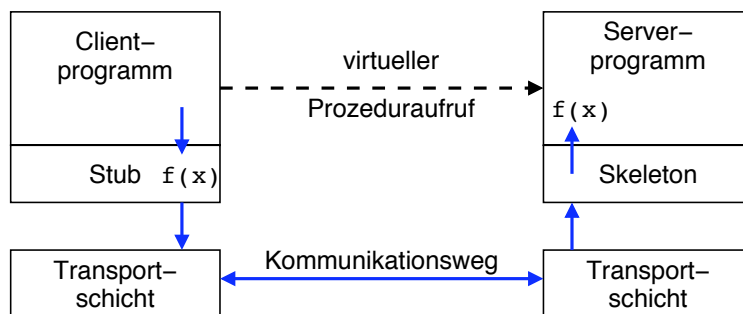
Entfernte Prozeduraufrufe: Grundlegende Fragestellungen

- Beschreibung einer Schnittstelle (z.B. mittels Beschreibungssprache)
 - Binden von Namen an Endpunkte (mittels Verzeichnisdienst)
 - Registrieren von Funktionen/Schnittstellen bzw. Objekten
- Prozeduraufrufe entsprechend
 - Verbindungsmanagement (z.B. Verwaltung von Pools von Verbindungen)
 - Serialisierung von Parametern und Rückgabewerten
 - Fehlerbehandlung (lokale und entfernte Fehler)

Rechnernetze

RMI Remote Method Invocation

- RMI: **Remote Method Invocation**
- Rollen: Server (bietet Schnittstelle), Client (nutzt Schnittstelle)
- Integration in Software durch **Skeletons** und **Stubs**
- Hilfsmittel: Verzeichnisdienst

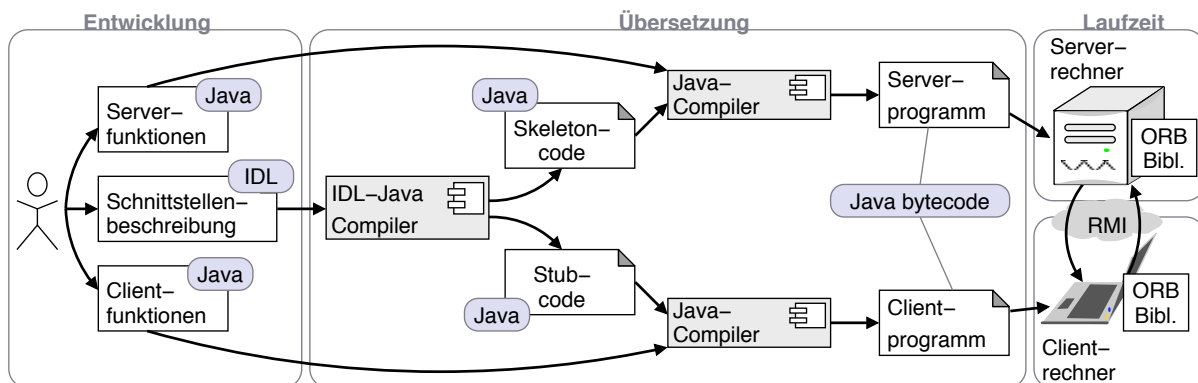


- Stubs/Skeletons: realisieren Ortstransparenz
- Client-/Serverprogramm: realisieren Dienst/Anwendung
- **Marshalling**: fertig stellen eines Aufrufs, so dass er versendet werden kann

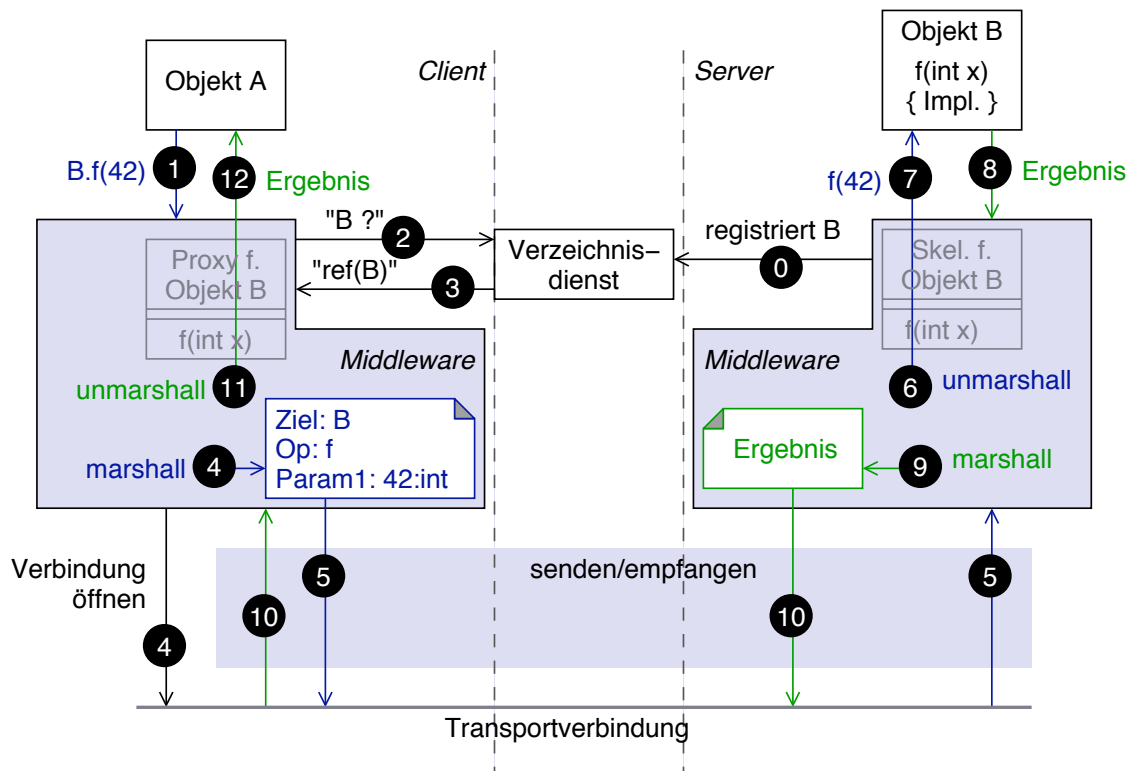
Gegeben seien eine Programmiersprache (P) und eine Schnittstellenbeschreibungssprache (IDL).

- 1 Programmierer schreibt Schnittstellenbeschreibung in IDL-Sprache.
- 2 IDL-Compiler erzeugt Quellcode in P, entsprechend der Schnittstellenbeschreibung.
- 3 Programmierer implementiert die Schnittstelle in P.
- 4 P-Compiler übersetzt generierten sowie geschriebenen Quellcode.
- 5 Binder/Linker verbindet Objektcode mit Middleware-Bibliotheken (z.B. mit CORBA-ORB)

Beispiel: Erstellung einer verteilten Java/CORBA-Anwendung



Rechnernetze



Rechnernetze

Internetdienste

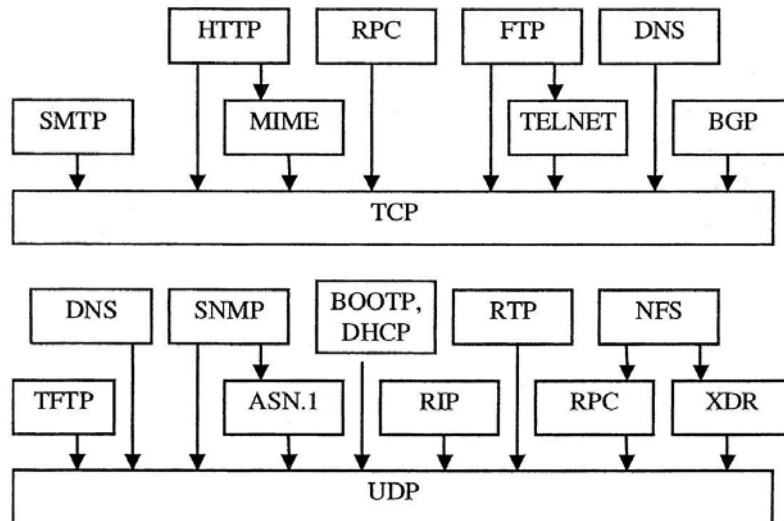
Aufgaben

- Ziele der Anforderungen
 - Unbeschwerte Nutzung eines gegebenen Dienstes
 - Dimensionierung des Netzes bzw. der Übertragungskanäle
 - Wahl von Transportverfahren
- Toleranz bezüglich Datenverlust
 - Audio-, Telephonie-, Video-Anwendungen, Spiele in gewissem Rahmen verlusttolerant
 - viele gängige Anwendungen erfordern verlustfreie, korrekte Übertragung (WWW, FTP, Email. . .)
- Benötigte Übertragungsrate
 - minimale Rate gefordert bei Multimedia-Anwendungen (z.B. Audio-/Video-Ströme; Zusammenhang mit Puffergrößen in Clientapplikation)
 - andere Anwendung haben „elastische“ Anforderungen; sie passen sich der verfügbaren Übertragungsrate an (z.B. Datei gegebener Größe wird schneller/langsamer übertragen)
- Verzögerung
 - maximale Verzögerung gefordert bei Echtzeitanwendungen (z.B. Telephonie, Spiele)

Anwendung	empfindlich bezüglich		
	Verlust	Übertragungsrate	Verzögerung
File Transfer	ja	elastisch	nein
E-Mail	ja	elastisch	nein
WWW	ja	elastisch (wenige Kbps)	nein
Audio/Video (Echtzeit)	tolerant	Audio: wenige Kbps – 1Mbps. Video: 10 Kbps bis 5Mbps	Ja, einige hundert Millisek.
Audio/Video (gespeichert)	tolerant	wie oben	Ja, wenige Sekunden
Interaktive Spiele	tolerant	wenige Kbps bis 10 KBps	Ja, einige hundert Millisek.
Finanz-anwendungen	ja	elastisch	ja und nein

Rechnernetze

- Auswahl von Transportdienst nach Dienstanforderungen
- TCP: zuverlässige Übertragung; höhere Kosten wegen Verbindungsaufbau, Quittungen etc
- UDP: nicht zuverlässig aber „sparsamer“ (kleinerer Header, verbindungslos)



DNS Domain Name System

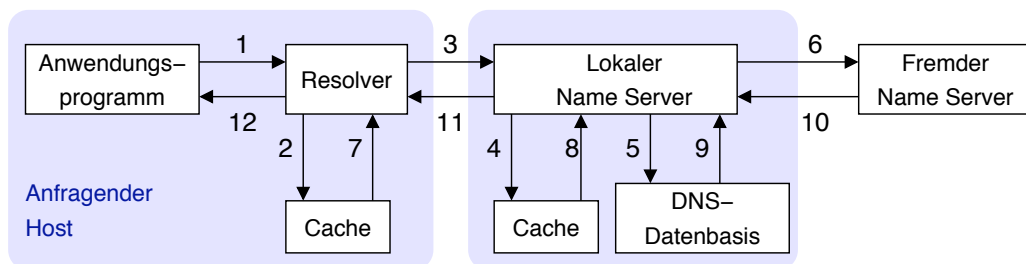
DNS

verteilte, in einer Hierarchie von Name Servern implementierte Datenbank und Protokoll der Anwendungsschicht zwischen Hosts und Name Servern

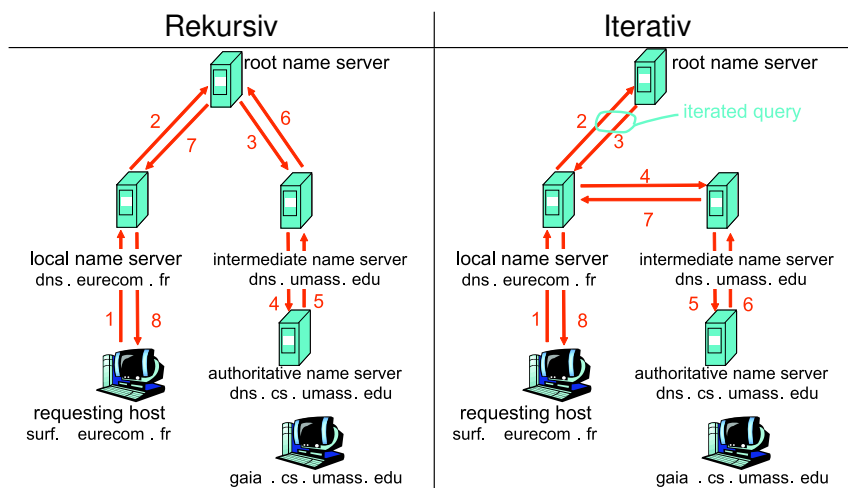
- Dienstmerkmale:
 - Abbildung von Host-Namen auf IP-Adressen
 - Host Aliasing (mnemonisch - kanonisch)
 - Mail Server Aliasing
 - Lastverteilung zw. replizierten Servern
- Funktionen eines DNS
 - Beantworten von Client-Anfragen
 - Austausch mit anderen DNS-Servern
- DNS ist kritisch für das Funktionieren des Internet
- Probleme: Bottleneck, Verkehrsvolumen, Entfernung, Pflege
- Hierarchie von DNS-Servern (wegen Skalierung) :
 - Lokaler Name Server, autoritativer Name Server, Root Name Server, vermittelnder Name Server

Rechnernetze

- Root Name Server
 - Nur wenige (ca. 1 Dutzend) Root Name Server weltweit, gut geschützt
 - löst Domännennamen auf, liefert Adresse eines autoritativen NS
- Authoritative Name Server: löst Host-Namen einer Domäne auf
 - „authoritative“: Antwort des Servers wird als richtig angenommen
 - für jede Zone: primärer DNS-Server, sekundäre Server (Ausfallsicherheit, Lastausgleich)
- Lokaler Name Server: Namensauflösung für Nutzer eines Betreibers
 - Host muss konfiguriert werden, lokalen NS zu kennen (mit IP-Adresse!)
 - Adresse des lokalen NS wird dem Host manuell oder mit DHCP mitgeteilt
- Resolver: Softwarekomponente zur Host-lokalen Namensauflösung
- DNS Software: meist *BIND* (Berkeley Internet Name Domain)



- Schritte
 - 1 Anfrage an Resolver (lokal auf Host)
 - 2 Nachschlagen im Cache. Erfolg → 7, 12
 - 3 Anfrage bei lokalem DNS-Server.
 - 4 Nachschlagen im Cache. Erfolg → 8, 12. Update Resolver-Cache.
 - 5 Nachschlagen in Datenbasis. Erfolg → 9, 11, 12.
 - 6 Anfrage an fremden DNS-Server. Antwort über 10, 11, 12.
- Inhalt der Antwort
 - gesuchte IP-Adresse, oder
 - Referenz auf DNS-Server, der den Namen auflösen kann, oder
 - „gibt es nicht“
- Antwort führt zur Auffrischung der Cache-Information
- Rekursiv: Name Server reichen Anfrage durch (→Last)
- Iterativ: Anfragen der Reihe nach



Rechnernetze

Resource Record (RR)

- Datensatz über einen Namen. Name Server liefert ihn als Antwort.
- 5-Tupel: Name, TTL, Class, Typ, Wert
- Typen (Auswahl)
 - A "Address": Name = Host, Wert = IP-Adresse (wichtigster Typ)
 - AAAA : wie 'A' für IPv6-Einträge
 - NS "Name Server": Name = Domain, Wert = Hostname eines autor. Servers
 - CNAME "Canonical Name": Wert = kanonischer Name für Alias Hostname
 - MX "Mail Exchange": Wert = Hostname eines Mailservers mit Aliasnamen
- TTL: Gültigkeitsdauer in Sekunden (Maß für Stabilität des RR)
- Class: Netztyp "IN" im Internet (selten anderer Wert. . .)

Zones

- nicht überlappende Bereiche des Namensraumes
 - Zuständigkeitsbereiche für Name Server
-
- Zonentransfer
 - Oft mehrere Name Server pro administrative Domäne: Primary NS, Secondary NS
 - Primary/Secondary NS bearbeiten Anfragen gleichgestellt (liefern autor. Antworten)
 - Konsistenzerhalt in der Datenbank der Name Server
 - Wartung einer zentralen Datei mit Resource Records
 - Unterscheidung alter/neuer Versionen mittels Seriennummer
 - Verteilung geänderter RRs mittels Zonentransfer
 - Secondary NS (Client) erfragt periodisch Änderungen beim Primary NS (Master)
 - Transportprotokoll
 - Anfragen: UDP-basiert
 - Anfragen zustandslos
 - Wiederholung möglich, falls Frage oder Antwort verloren gehen
 - Verzicht auf Verbindungsaufbau → Performanz
 - Zonentransfer: TCP-basiert
 - Zuverlässigkeit von Bedeutung; größeres Datenvolumen als eine Anfrage
 - findet selten statt (im Vergleich zu Anfragen)

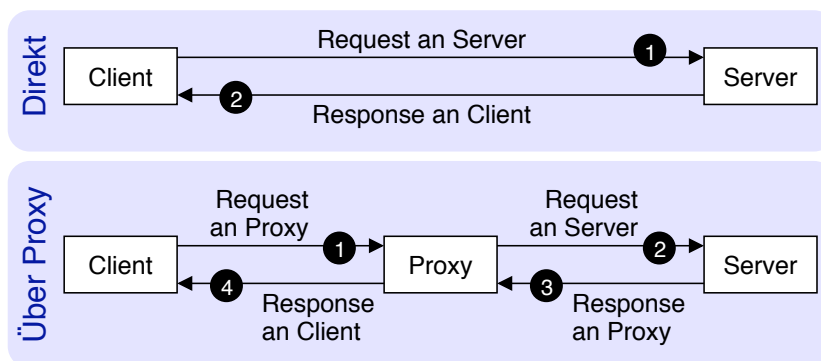
Rechnernetze

Mail

- Standard für Textnachrichten, spezifiziert in RFC 822
- Nachricht besteht aus Kopf/Header, Body sowie Abschlusszeile
- Header-Zeilen (siehe auch Beispiel in Kapitel 3)
 - To, From, Subject
 - CC, BCC ([Blind] Carbon Copy)
 - Reply-To: Emailadresse, die für Antwort benutzt werden sollte
 - Message-Id: identifiziert eine Emailnachricht in späterer Kommunikation
 - In-Reply-To, References: Verweise auf Message-Ids von Emailnachrichten
 - Received: wird von jedem vermittelnden MTA dem Header hinzugefügt
- Body
 - Nutzdaten („Brief“); nur ASCII-Zeichen zulässig
 - Letzte Zeile markiert Nachrichtenende; sie enthält nur einen Punkt ‘.’
- Erweiterung zum Transport von Multimedia-Daten
 - MIME: multimedia mail extension, RFC 2045, 2056
 - zusätzliche Information im Header → Angabe des MIME content type

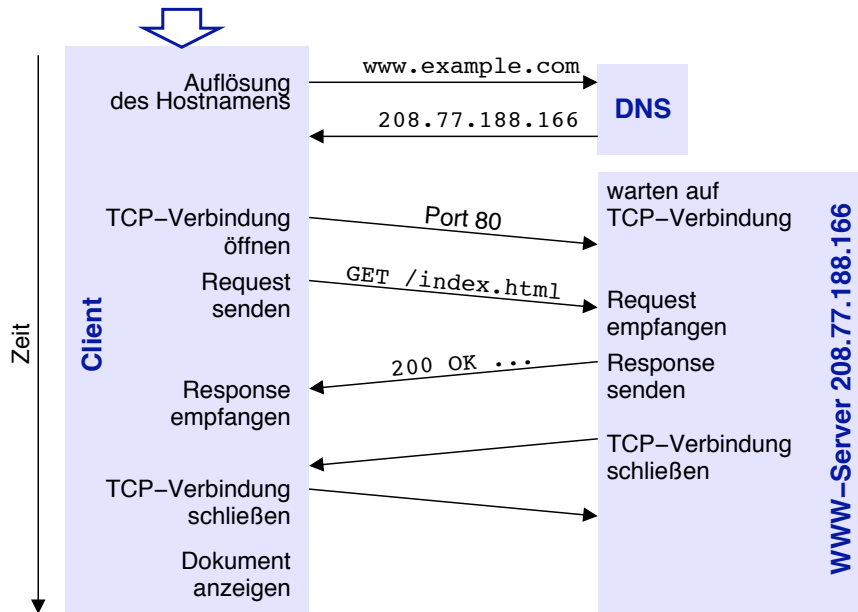
WWW und HTTP

- Protokoll zum Transport von Anfragen/Objekten zwischen Browser, Server und Proxy (Zwischensystem)
- TCP-basiert, Port 80 (Proxy: typischerweise Port 8080)
- HTTP ist zustandslos, pull-orientiert, unterstützt bidirektionale Übertragung und Caches im Client bzw. Proxy
- HTTP/1.1 spezifiziert in RFC 2068, 2616



Rechnernetze

Nutzereingabe in Browser
`http://www.example.com/`



HTTP als zustandsloses Protokoll

TEAM

- Vorteile: einfach → fehlerunempfindlich
- Nachteil: viele Dienste benötigen Zustandshaltung
 - Dienstsitzung besteht aus **mehreren Schritten** (z.B. Buchung einer Fahrkarte)
 - Request/Response-Paar entspricht einem einzigen Schritt
 - Dienststatus muß **über alle Schritte** erhalten werden
- Abhilfe (zum Standard erhobene Notlösungen)
 - Statusvariablen **in URL**, werden per GET-Methode übertragen
 - Status wird **in Daten der POST-Methode** übertragen
 - Cookies: **clientseitige Speicherung** einer (kleinen) Datenstruktur
 - Entfremdung von HTTPS (Benutzerauthentifizierung bestimmt Sitzung)
 - SessionID: Vergabe eindeutiger Kennung für eine Dienstsitzung (oft in URL); Status **serverseitig gespeichert**.

Rechnernetze

Ziele: schnelle Anzeige von Webseiten; geringe Netzlast

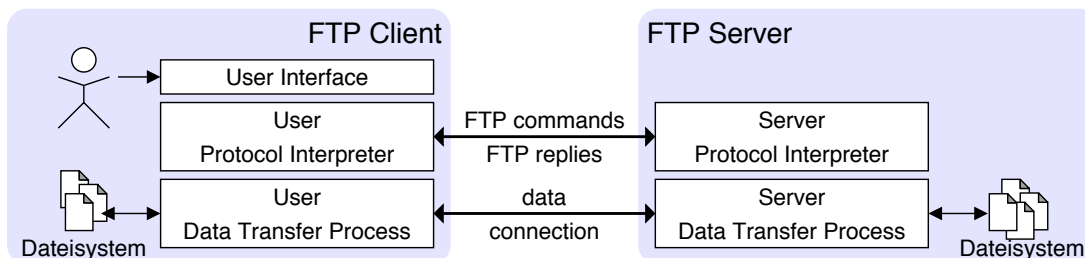
Einfachster Fall

- Eine TCP-Verbindung zum Server pro Request/Response-Paar
- Nach Request wird auf Response gewartet.
- Client ruft Objekt jedes Mal vollständig vom Server ab.
- Verbindung wird nach Interaktion geschlossen.

Maßnahmen zur Leistungssteigerung

- Mehrere **gleichzeitige Verbindungen**: Parallelisierung der Anfragen
- Persistente Verbindung: Verbindung wird für mehrere Anfragen wieder benutzt → Einsparung des Verbindungsaufbaus ("Connection: **keepalive**")
- Pipelining: Client schickt **mehrere Requests nacheinander**; Server schickt entsprechende Responses.
- Caching: Client verwaltet lokales **Cache** jüngst abgerufener Objekte.
- Conditional GET: Objekt wird **nur übertragen, falls neuer** als Cache-Version.
- Caching proxy: Anfragen werden durch Proxy geleitet. **Proxy verwaltet Cache** → gemeinsames Nutzen des Cache

FTP



- Übertragung von Dateien zwischen zwei Hosts (RFC 959)
- *control channel* (TCP, Port 21): Befehle, Antworten
 - out-of-band Befehle (→ nicht verwechselbar mit Nutzdaten)
- *data connection* (TCP): Übertragung von Dateien (bidirektional)
 - nur für Dateiübertragung geöffnet
 - *aktives* FTP: Server öffnet data connection
 - *passives* FTP: Client öffnet data connection

Internet Management

Motivation

Warum Management ?

- ☐ Bisher nur Nutzungsfunktionalität in Form von Protokollen und Diensten besprochen
- ☐ Ein Netz und seine Dienste müssen aber mit konkreten Ressourcen, mit steuerbaren Dienstgütern, in konkreten Organisationen und mit belastbaren Zielvereinbarungen betrieben werden.
- ☐ Managementobjekte sind u.a. Verbindungen (auf jeder Schicht), Koppellemente auf verschiedenen Schichten wie z.B. Hubs, Bridges, Switches, Router, Gateways, Multiplexer, ferner Protokollinstanzen, Endsysteme, Softwarekomponenten, Dienste, Anwendungen
- ☐ Zielvorgaben: Verfügbarkeit, Zuverlässigkeit, Sicherheit, Durchsatz, Reaktionszeiten, Lastausgleich, Kosten

Arten von Management

Netzmanagementdienste

- ☐ Benutzerverwaltung, Abrechnungsmanagement
- ☐ Sicherheitsmanagement
(Bedrohungsanalyse, Sicherheitsmechanismen, Verschlüsselung, Authentifizierung, Zertifizierung)
- ☐ Fehlermanagement
(Symptomerfassung, Eventkorrelation, Fehlerdiagnose, Fehlerbehebung, TT-Systeme)
- ☐ Konfigurationsmanagement
(Generieren und Installieren von Systemen, Parameterfestlegung, Statusüberwachung, Versionsverwaltung, SW-Verteilung, Topologieplanung)
- ☐ Leistungsmanagement
(Leistungsmessung, QoS-Parameter, Engpassanalysen, Auslastung, Kapazitätsplanung)
- ☐ Ressourcenmanagement (z.B. Bandbreiten, Wege)